Querying Office Systems about Document Roles

A. Celentano[1,2], M.G. Fugini[1,3], S. Pozzi[2]
(1) Università di Brescia, Via Valotti 9, I-25100 Brescia, Italy
(2) CEFRIEL, Via Emanueli 15, I-20126 Milano, Italy
(3) Politecnico di Milano, Piazza L. da Vinci 32, I-20132 Milano, Italy

ABSTRACT

This paper describes the architecture of a document retrieval system integrating classical IR features with knowledge about the procedural and application context where documents are used. The paper focuses on the query language that allows the user to pose queries involving the analysis of both the semantic network where procedures, office agents, and events of the office context are represented as elements accessing, modifying, filing, manipulating document, and the document contents, i.e. their text. The coupling of the query system with a browser tool is also discussed. The system relies on a knowledge representation model for document and document roles developed in previous phases of the research.

1. INTRODUCTION

In the paper "Knowledge Based Retrieval of Office Documents", presented at the SIGIR '90 Conference [Celentano 90a], we discussed the classification and retrieval of office documents through a knowledge based description of their role in the office environment. The paper introduced a research project carried on at CEFRIEL and illustrated the overall document model and the principles of operations of the retrieval system. As the project was going on, further developments led to the design of a set of tools for the classification and management of documents: the basic tool is a browser supporting advanced navigation functionality along the semantic network which represents the knowledge about document roles [Celentano 90b].

In this paper the project issues related to the automatic retrieval of documents are discussed: they include the system architecture, integrating the user workstations with a centralized text-based information retrieval server, the query language, which supports the semantic model of documents together with conceptual structuring and text indexation, and the integration between the browser and the query processor.

The query language integrates the following features:

- it allows the user to ask for properties related to the conceptual structure of documents;
- it allows to retrieve documents using traditional keyword-based and index-based techniques;
- it allows to pose queries about document roles, using knowledge on the relationships holding among documents of the office environment. These queries cannot be supported if the content alone is represented, but an explicit representation of both the document processing modes and the reasons for the existence of documents within a given application context is needed.

In spite of the broad scope provided for queries, the retrieval system remains strongly oriented to documents, which are the elements that can be directly addressed as the target of a query. In fact, the office environment where documents are used is represented in the retrieval model to the extent necessary to support queries about the document environment and life; as opposite, in Office Information Systems [COIS 90] also procedures and procedural flows are explicitly represented.

The reader is referred to the above cited papers for the presentation of the rationale for modeling office knowledge in document retrieval systems, the document model, the knowledge representation, and the navigation in the document base. However, to make this discussion self-contained, a brief description of the document retrieval model is surveyed in Section 2. Section 3 illustrates the system architecture and the relationships between the browser and the query processor. In Section 4 the proposed query language is described. The status of the implementation, and the current research activities are described in Sections 5 and 6.

The examples given in the paper refer to a test case currently used for experimenting the development of the system prototype. The test case regards the banking application domain, in particular the documents managed in the process of granting loans to customers by a bank credit department. It has been derived from the TODOS Esprit I

Project [Pernici 90]; it is a real, medium-size application, useful to test the power of the document modeling approach because it has a significant number of document types and a relevant procedural complexity.

2. MODELLING OFFICE KNOWLEDGE FOR DOCUMENT RETRIEVAL

The document model defined in this project is oriented to classification, filing and retrieval of office documents. In particular we made a distinction between a conceptual document model and a document retrieval model.

The former regards the semantic, logical, and layout representation of documents. Each document is modelled by means of a *conceptual structure*, that is, a structured collection of properties whose semantics, rather than textual appearance, is relevant [Thanos 90]. Conceptual documents are used in classification and retrieval activities, in order to take into accounts the semantic aspects which are not directly and explicitly contained in the document text. Conceptual documents are instances of classes, which define equivalences among documents having the same structure and meaning. An *Is-a* class hierarchy defines refinement and generalization relationships according to the object oriented paradigm.

The document retrieval model enriches the conceptual model with the explicit representation of both the documents role within the office applications, and their dependencies from the domain knowledge, i.e. rules, regulations, habits, laws which are part of the motivation of existence of documents. *Procedures*, *agents* and *events* are introduced as individual classes for representing the tasks in which documents are used, the actors performing the tasks, and the temporal events triggering the activities respectively. They are represented at the minimum level of detail required to establish relationships among documents and they are identified through class instantiation to distinguish among different occurrences of the same type of task, role and temporal event.

The document retrieval model entities (i.e. documents, procedures, agents and events) are connected by *links* describing:

- document relationships, e.g. a document *belongs to* a dossier;
- causal dependencies, e.g. an event *triggers* a procedure which *creates* a document ;
- organizational dependencies, e.g. a person *executes* a procedure which *modifies* a document.

Document relationships are modelled by links called *document links*, relationships between procedures and documents are modelled by *process links*, while *activity links* relate events to procedures.

The model entities and the links established among them, define a semantic network, where nodes represent the model entities and the arcs constitute the links; it can be seen as composed of two layers, a class and an instance layer. Moreover for each link a named inverse link is defined, to enable the navigation in both directions. The semantic network describes the operational knowledge in the office. Figure 1 illustrates a part of the class layer of such a semantic net for the test case currently used in the prototype development; the fragment regards the credit contract preparation: collecting customer information, preparing the customer dossier, drafting the offer, until the contract signing.

The explicit representation of procedural knowledge supports the resolution of the following query types (some of them refer to items not represented in Figure 1):

- "Retrieve all the customer information letters archived by the secretary Susan";
- "Which documents have been submitted by Foo&Co. for obtaining a credit?"
- "Retrieve the reports prepared after discussing credit grant opportunities in January 1991"

These queries are examples of *operation-oriented* queries performed by the user of the system, to retrieve document instances needed to perform specific office tasks. The system addresses also *guidance-oriented* queries posted by the end-user to get assistance; examples of such queries are the following ones:

- "Which documents need to be sent to the customer after the preparation of a credit offer letter?"
- "Which documents are still missing from the dossier of Foo&Co?"

Guidance-oriented queries are also the basis for setting-up, customizing, and maintaining the retrieval system. Such activities are performed by the System Administrator.

Concepts and rules are elements of the domain knowledge of the document retrieval model that characterize the application context of the office. Examples of application contexts are legal, technical, staff-support and administration offices. In each context, sets of concepts and dependencies peculiar of the domain can be outlined. Due to the fact that they often involve more than a couple of entities, and that they do not establish direct relationships, but rather define the consistency of a set of entities, dependencies are modelled as logic rules better than as links.

Within the domain of banking offices, for example, concepts are *loan*, *warrantor*, *amount*. These concepts can be used, in the form of predicates about the status of a specific loan granting case, and about the related documents, to interpret user queries by providing a synonym mechanism for expressing complex retrieval conditions on the content and relationships between documents. Rules express the reasons why documents are processed in a given way; moreover, they are used to express laws and regulations that define mutual relationships among documents, which are not expressed directly in the documents contents or in the procedural descriptions. For example, using the above referenced concepts of the loan granting domain, the following rules (in Prolog-style notation) define when warrants are required:

```
needs_warrant(Loan_id) :-
        loan(Loan_id,Customer),
        company(Customer)
        amount(Loan_id,Money),
        greater(Money,10000)
```

```
warranted(Loan_id) :-
    loan(Loan_id,Customer),
    needs_warrant(Loan-id),
    warrantor(Loan_id,W1),
    warrantor(Loan_id,W2),
    W1 ≠ W2
```

The former rule states that a warrant is needed if the loan amount granted to a Company (as opposite to a private customer) exceeds $10,000; the latter rule states that if the loan must be warranted, two warrantors are required. For the retrieval system, this rule implies the existence of two warrant letters associated to the dossier of the credit recipient when this recipient is a company; therefore, the implicit relationship between the *Customer-Dossier* document and *Warrant-Letter* document is considered through a knowledge inference process. The reader is referred to [Celentano 91] for a more thorough discussion about the definition and use of domain knowledge.

3. THE ARCHITECTURE OF THE SYSTEM

The system architecture design has been guided by the following considerations:

- in most office environments it is common that documents are archived and retrieved by means of "traditional" IR tools [Salton 88]. They provide effective and efficient access to large amounts of texts, enabling the user to pose queries related to the content of the documents. However, they do not consider the role of the documents in the office, which is a key concept of the proposed system. The architecture should enable to blend a knowledge based approach with traditional IR techniques, resulting in a significantly more effective system;
- the main features characterizing the office environment are the distribution and the complexity of the performed activities. Two facts stem from this consideration. Firstly, the construction of the proposed document retrieval model will in general require the collaboration of several designers, to tackle the complexity of the office environments. The system architecture should provide a basis to build tools enabling effective communications among the work group [CSCW 90]. Secondly, the defined document model should reside in a centralized repository, enabling the access of several user workstations. This solution eases the process of updating the model and it avoids useless replication of data on the different user workstations.

The architecture of the system is based on the client-server paradigm, where the user workstations act as clients, interfacing with two servers, namely the *document server* and the *model server*. A text based IR tool runs on the document server: it manages the document base (i.e. the collection of the inserted documents) thus providing a mean to solve queries related to the documents content. On the other hand a database management system runs on the model server: it acts as a static repository of both the entities defined in the model (i.e. classes and instances) and the several rules exploited by the system during the insertion and browsing phases. Moreover the centralized data repository can be exploited in the future to define communication mechanisms among the office designers. A reference obviously exists between the document and the model bases, in order to enable the retrieval of the appropriate documents.

The architecture of the system can be split in two fundamental subsystems, which also refers to the two different functional phases, namely:

- the filing phase and
- the retrieval phase.

Figure 2 illustrates the functional partitions of the system.

The filing system

The general task of this subsystem is the acquisition and classification of documents. These processes are carried out by the following modules:

- the Class Specification Module (CSM);
- the Instance AcQuisition module (IAQ);
- the CLASSifier module (CLASS);
- the Semantic Net Interface module (SNI);
- the Information Retrieval System Interface module (IRSI);
- the NeTwork Service module (NTS);
- the DataBase Management System module (DBMS) and
- the Information Retrieval System module (IRS).

CSM enables the user to define class structures. The user defines document classes by means of a window-based user interface, designed on top of a *document definition language*. During the class definition process the user can either browse or query the document semantic network, for instance to obtain information about the existing classes and links. A document class definition is inserted in the model base, residing on a remote server. To connect with the remote model base, CSM exploits the functionalities offered by SNI.

IAQ extracts the conceptual representation of the document from the document text. In particular a document is represented as a labelled tree, where the labels are the name of the conceptual fields. The conceptual representation is then passed to CLASS, whose main scope is the classification of the inserted document. It is likely that such representation will not exactly be isomorphic with an existing class definition, due to the fact that either IAQ has not fully recognized the conceptual structure of the document or that the document class corresponding to the inserted instance does not actually exist. A set of rules residing in the *classification rule base* helps CLASS performing the classification process, i.e. finding the class (or classes) name(s) whose conceptual structure best matches the structure of the inserted instance. A ranked list of the class names resulting from the classification attempt is presented to the user: he will choose the class which best suits his requirements.

Moreover CLASS plugs the instance conceptual structure into the document instances network, instantiating the appropriate links. To perform such process, CLASS exploits the *insertion rules*. Finally, the document textual representation is stored in the document base. To connect with both the remote model base and the remote document base CLASS exploits the functionalities offered by SNI and

IRSI.

SNI interfaces with the model server. In the filing phase, SNI features are exploited in order to insert into the model base the conceptual structure of both document classes and document instances. DBMS manages the model and the rule bases. Such databases act as repository of objects which become alive when they are transferred in the user workstation modules.

IRSI interfaces with IRS. In the filing phase it enables a document instance to be stored and indexed by means of IRS.

NTS handles the transfer of query commands and data files from/to SNI and IRSI of the user workstation to/from DBMS and IRS resident on the model and document server respectively.

The retrieval system

The general task of this sub-system is the retrieving of both document classes and instances from the model base and the document base respectively. The functions involved in the retrieval phase are carried out by the following modules:

- the User Interface Management System module (UIMS), split into the BROWser (BROW) and the QUery System (QUS) modules;
- the Semantic Net Interface module (SNI);
- the Information Retrieval System Interface module (IRSI);
- the NeTwork Service module (NTS);
- the DataBase Management System module (DBMS) and
- the Information Retrieval System module (IRS).

A document can be retrieved in two ways: by means of an interactive browsing activity, performed on both the classes and instances of the semantic network, and by means of a query formulated in the defined query language. UIMS offers a window-based man machine interface consisting in both the browsing and the query system. It highly integrates the two subsystems, allowing the user to interleave browsing and querying operations.

Browsing is organized in sessions, that is, sequences of navigation paths along the links of the network. The nodes traversed in a browsing session can be saved and referred to as part of a query, in order to restrict the automatic retrieval to a selected partition of the document base.

The browser offers several hypertext-like tools which can be exploited to navigate through the documents model. The user browses the document base, visualizes the semantic network describing the application document roles, and, basing on the result of the visualization, decides either to further explore the network, or to zoom into some nodes for further information, or to enter new search parameters for restricting the exploration to portions of the network.

The users of the document retrieval system can browse both the class and the instance layers of the semantic net. The target of a class browsing process is a set of document types, for example, the documents needed from the customer who applies for a credit. By moving to the instance layer, the specific occurrences of documents can be retrieved. A detailed description of a sample user session can be found in [Celentano 90b].

BROW exploits also navigation rules, to infer the existence of links which have not been explicitly created by the user. The user interacts with the query system by means of a menu-based interface, which has been designed on top of the defined query language. QUS breaks up a query into the components involving the semantic network, the conceptual structure and the text analysis. The query components involving the analysis of both the semantic network and the conceptual structure are resolved by means of DBMS. On the other hand, query components involving the analysis of the document text are resolved by means of IRS. QUS receives the results of the execution of the query components and performs set operations on it, to present the result to the user.

4. THE QUERY LANGUAGE

Several approaches to the characterization of the documents meaning through the description of a "conceptual" layer have been discussed in the literature (see for example the Esprit project MULTOS [Thanos 90]). Our project shares with them the characterization of the document concepts by formalizing its structure; therefore the aspects of the query language related to the conceptual component will be discussed to a less extent while the presentation will focus on the exploitation of the operational and domain knowledge.

The query language is based on three different facets of documents identification:

- the text contents of the documents;
- the structured data which describe the documents static meaning (i.e. the conceptual structure);
- the environment in which the documents are embedded, that is the relationships holding among documents and among documents and procedures using them.

IR traditionally addresses the first topic. Systems which use structured description of documents (i.e., logical or conceptual schemas) together with free text contents also exist. The structure is helpful in identifying the meaning of a document in a more precise way, since the conceptual structure can be formally described. The query language aims at integrating existing approaches with more advanced functions about knowledge representation.

4.1 THE QUERY FORMULATION

A query is defined by the following general structure, in which boldface denotes keywords, italic denotes fragments defined in detail later, plain text denotes placeholders for specific terminal items, parentheses identify syntactic precedences, brackets and braces denote optional and repeated elements, and the vertical bar divides alternative choices:

retrieve class_name (**types** | **instances**)
[**in** *partition*]
[**with** *conceptual_description*]
[**environment** *environment_description*]
[**contents** *contents_description*]

We defer the discussion of the "**in** *partition*" clause to a later point in this paper, since it involves also the relationship between the query processor and the browser.

The target of the query: the *retrieve* clause

The output of the query can be a set of document instances, or a set of types; the latter case refers to queries directed to exploring the structure of the office document model, and occurs when the user performs a guidance oriented search, as previously discussed.

The name of a class is required to restrict the search to a subset of the defined document classes; while a user experienced in the application field can suitably select a sub-domain on which to operate, a novice user can start with a broader range, for instance indicating in the **retrieve** clause the the root class *document*, if he is not able to indicate a more specific request.

If the query target is a set of instances, the instances of both the specified class and its subclasses are looked for. If the query target is a set of types, the query returns the set of the most specific subclasses which satisfy the query clauses.

The *with* clause

The **with** clause describes the static meaning of the requested documents, in terms of their conceptual components. It is composed of a sequence of predicates on the attributes of the document structure, in the form of boolean expressions:

field_name relational_operator value.

Expressions can be joined by **and** and **or** operators, and parentheses can be used to modify the evaluation order. For instance the following query

retrieve Solicitation_letter **instances**
with (Receiver.Name = "Foo&Co." **or**
Receiver.Name = "Such&Such")
and date >= "3/1/1991"

retrieves solicitation letters sent to the customers Foo&Co. and Such&Such on or after march 1st 1991.

If the query target is a set of types, predicates on values are meaningless, and the clause can only predicate the presence of conceptual components by referencing their names. The following query

retrieve document **types**
with offer_validity

retrieves only document classes having a conceptual component called *offer_validity*.

The *environment* clause

The **environment** clause enables to state the relationships which must hold between the searched document and other office entities. Such knowledge is coded in terms of links of the semantic network which models the document base: by means of the **environment** clause the user expresses a predicate on the role of the document in the office system. It has the following general structure:

environment
{ link_name (node_name | { node_list } |
partition)
[**subset** conceptual_description] }+

where *link_name* is the name of a link in the semantic network; *node_list* is a list of node names separated by commas; *node_name* is the name of a class or of an instance (in the form *class-name:instance-name*) of the semantic network; *partition* is a specification similar to that used in the "**in** *partition*" clause, therefore it will be described later. The **subset** clause duplicates the usage of the **with** clause, but applies to the referenced nodes rather than to the requested documents. The path specification can be iterated, resulting in compound dependencies.

The **environment** clause identifies as candidate results of the query the types or instances which are connected, through the named link, with the specified node or set of nodes. According to the target of the query (i.e. types or instances) the connected nodes can be types or instances, with the following constraints:

- if the query target is a set of types, the referenced nodes must be classes;
- if the query target is a set of instances, the referenced nodes can be instances or classes; if they are class names, they identify the set of instances of that class;

Examples of environment specifications are the following ones (please refer to Figure 1):

a. the documents archived in the dossier of customer Foo&Co.:

retrieve document **instances**
environment belongs_to Customer_Dossier:Foo

b. the documents (types) needed to complete the dossier of a customer:

retrieve document **types**
environment is_input_of
Procedure:Customer_Dossier_Completion

c. the documents that were handled by Schultz in some way:

retrieve document **instances**
environment process_link Procedure
executed_by
Agent:Schultz

d. the documents that *can be* handled by Schultz in some way:

retrieve document **types**
environment process_link Procedure
executed_by
Agent:Schultz

In the last two queries, it is necessary to note that, even if not depicted in Figure 1, the nodes of the network are connected by pairs of links, establishing direct and inverse relationships. Therefore the *process_link* referenced in the queries *c* and *d* stands both for documents which are output by a generic procedure, and for documents which are consumed as inputs, thereby identifying in a correct way the concept of "handling".

The *subset* clause

The **subset** clause excludes from the set of nodes referenced

in the **environment** clause, the nodes which do not satisfy the conceptual description predicates. The formulation of the clause is subjected to the same constraints defined for the **with** clause, that is:

- if the query target is a set of types, it must predicate the presence or absence of components, but it cannot predicate about values;
- if the query target is a set of instances, it must predicate about values;
- as a special case, if a single node instance is referenced in the environment clause, the subset clause acts as an "only if" constraint, which accepts or rejects the referenced instance according to the values of its conceptual components.

As an example, the query *a* above can also be formulated in this way:

retrieve document **instances**
environment belongs_to Customer_Dossier
subset Name = "Foo&Co."

The *contents* clause

The **contents** clause predicates about the textual part of the documents, bridging the conceptual and semantic description to the actual contents. It is a usual information retrieval query, that is, its lexicon, syntax and operational structure are defined according the actual IR engine to which this system is interfaced, and will not be detailed here.

The *in* clause

Retrieval is seldom a one-shot process; the specification of the required documents and the sharp identification of their relevance in the returned set are the result of a refinement process involving several accesses to the document and knowledge bases. The **in** clause defines a partition in the document set, identified by a symbolic name which denotes the result of a previous query, or the marking of selected nodes during browsing; the query is then executed by taking this partition as if it where the whole document base, and its associated semantic net. If specified as part of the **environment** clause, the partition is used to identify the set of nodes which are the target of the named links.

As an example, suppose that the user, while browsing the document and knowledge base associated to the fragment of semantic net illustrated in Figure 1, has marked as relevant the nodes of the network excluding the procedure *Contract_Signature* and the associated documents, *Customer_Contract* and *Final_Offer*, identifying this set with the name *Preparation_Phase*. The execution of the query

retrieve documents **instances**
in Preparation_Phase
environment process_link Procedure executed_by Agent:Schultz

retrieves the documents of type *Draft_Offer* and *Report*, but not the documents of type *Customer_Contract* and *Final_Offer*.

4.2 THE DOMAIN INFERENCE

The identification of the relationships between documents and the entities of the office domain are described in the **environment** clause, explicitly indicating the links. The burden of tracing a long navigation path can be reduced, and the knowledge support provided by the system can be improved, by of the definition of *virtual links*. A virtual link is a kind of link which does not explicitly exists in the model network, rather its presence can be inferred by topological and domain rules which maintain knowledge about complex relationships. As a simple example, consider the query, already discussed:

"Retrieve the documents that were handled by Schultz in some way"

which was formally stated by a two steps environment clause:

retrieve document **instances**
environment process_link Procedure executed_by Agent:Schultz

The concept of document handling by an office agent is bound to the execution of a number of different tasks; in its widest formulation, it can be described by the following topological rule:

```
handled(Doc,Agent) :- is_instance(L,process_link),
    is_instance(P,procedure),
    s_instance(E,executed_by),
    links(Doc,P,L),
    links(P,Agent,E).
```

with obvious meaning. By rewriting the left hand side of the rule with the infix notation

Doc handled_by Agent

the query above can be written in the following way:

retrieve document **instances**
environment handled_by Agent:Schultz

The query execution requires a forward resolution engine to fire the rule for identifying which real links need to be explored.

5. IMPLEMENTATION NOTES

The system is implemented on a heterogeneous platform. The User Workstation is a MS-DOS/MS-Windows 80386 system running GoldWorksII[1], which is an expert system development environment based on GCLisp, exploiting object orientation, frame based techniques, and user controllable production rules to represent knowledge.

The Model and Document Server are implemented in a Unix environment; Oracle[2] is used as DBMS, and BRS/Search is used as the text oriented IRS. The communication between the two environments is managed, by means of the NTS module, using the TCP/IP protocol.

---

[1]GoldWorksII is a Trademark of Gold Hill Inc.
[2]Oracle is a TradeMark of Oracle Corp

Currently, the implementation effort is devoted to the integration of the User Workstation modules; in the UIMS, the browser (BROW), formerly developed as a first prototype with a different Object Oriented tool (Kool/Aida[3]), has been ported to the GoldWorksII environment and interfaced to the semantic network with the NSI module. The parsers for the Class Specification and Query Languages and the query processor component related to the resolution of the **environment** clause on the semantic network has been implemented. The network services have been defined and built as general purpose interpreters of commands for task to task communication; the DBMS and IRS interface functions have been designed.

6. FUTURE DIRECTIONS

The current research effort focuses on the formalization and use of domain knowledge. Domain concepts and rules increase the power of the retrieval system by allowing the system to trace relationships deriving implicitly from the application domain. To this aim, the query language should also include "concepts" of the domain and automatically derive the query processing actions to be undertaken to exploit that concepts.

Further work needs to be done in the project on the definition of an inferential engine for domain rules and for extending the query language to include domain concepts. To give an idea of the kind of knowledge that is required to take into account domain rules, consider the following statement:

> "If a customer has not answered a credit offer letter within two weeks from the offer forwarding, he should be solicited through a letter signed by the head of the credit department"

A guidance oriented search, aimed at understanding the roles of documents in the office procedures, could then use this knowledge to answer queries like the following ones:

> "Which documents should be prepared to solicit a customer to answer an offer?"

As another example, consider the following query:

> "Retrieve the Balance Sheets in the dossier of customer Foo&Co. which have been archived by Schultz and which are coming from the Chamber of Commerce".

Here, the target is a set of document instances; the fact that the Balance Sheets come from the Chamber of Commerce is modelled through domain rules saying that:

> "The Balance Sheets regarding customers are the collection of sheets provided by the Chamber of Commerce, by the customer itself, by other departments and branches of the Bank. Some of these can be missing, e.g., if the customer had no previous relationship with the Bank".

The amount of knowledge contained in this definition is large compared with the previous examples. The types of documents provided by the different organizations, the form in which they are drawn, and the amount of information contained in them, depend on several conditions about the company status, its kind of business, the existence of commercial operations with foreign countries, and so on; a predicate based representation requires deep understanding of the underlying laws and regulations [Kowalsky 86]

[3]Kool and Aida are Trademark of Bull Corp.

ACKNOWLEDGMENTS

We would like to thank the persons involved in the research activity of the Office Automation area at CEFRIEL. Namely, Dr. Paola Gattoni from Bull HN provided support to the GoldWorksII programming environment. The students of the Office Automation area contributed to the refinement of the model and implemented the prototype described in the paper. Dr. Luca Passerini, from S.G.S. Informatica, and Dr. Mauro Lazzaretto developed the NTS modules.

REFERENCES

[Celentano 90a] A. Celentano, M.G. Fugini, S. Pozzi, "Knowledge-based retrieval of office documents", in Proc. 13th ACM SIGIR Conference on Information Retrieval, Brussels, September 1990.

[Celentano 90b] A. Celentano, M.G. Fugini, S. Pozzi, "Document retrieval in office environment: knowledge modelling and browsing", CEFRIEL Technical Report RT 90028, October 1990.

[Celentano 91] A. Celentano, M.G. Fugini, S. Pozzi, "Expert System Support for Classification and Retrieval of Office Documents", Cefriel Technical Report, March 1991.

[COIS 90] Proceeding of ACM-IEEE Conference on Office Information Systems, Boston, MA, April 1990.

[CSCW 90] Proceedings of Conference on Computer Supported Cooperative Work, Los Angeles, 1990.

[Kowalsky 86] R.A. Kowalsky, M. Sergot, "The Use of Logical Models in Legal Problem Solving", Dept. of Computing, Imperial College, London, 1986.

[Pernici 90] B. Pernici, C. Rolland (Ed.s), "Automatic Tools for Designing Office Information Systems", Springer-Verlag, October 1990.

[Salton 88] G. Salton, M.J. McGill, "Introduction to Modern Information Retrieval", McGraw-Hill, 1989.

[Thanos 90] C. Thanos (Ed.), "Multimedia Office Filing: the MULTOS Approach", North-Holland, 1990.

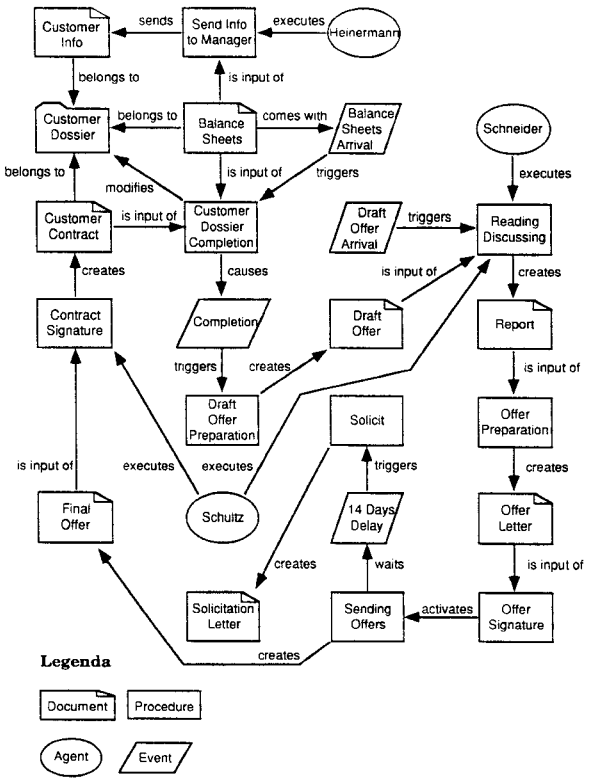

Figure 1. A fragment of the test case semantic net.

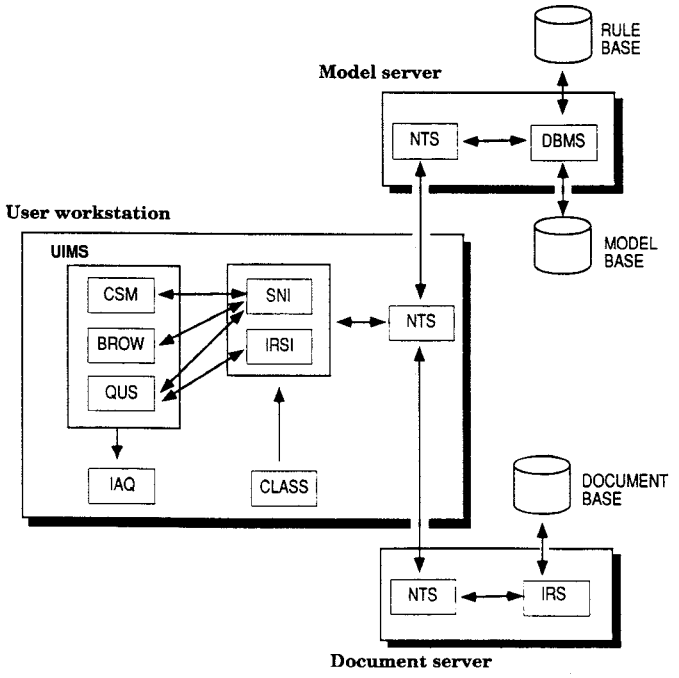

Figure 2. The architecture of the system

APPENDIX: THE GRAMMAR OF THE QUERY LANGUAGE

The grammar is expressed in a yacc-like notation, but a number of details have been omitted for readability.

Elements ending in *_name* , *_identifier* or *_operator* are syntactic terminal symbols, refined at the lexical level in obvious way; *value* denotes a constant value of one of the basic types defined for the conceptual components of the structured representation of documents; *domain_predicate* denotes a predicate of the domain knowledge, whose arguments are nodes and conceptual components of the semantic model. The precedence among the operators and the use of partentheses are not evidenced.

query : **retrieve** class_name objects
partition_part
conceptual_part
environment_part
contents_part

objects : **types**
| **instances**

partition_part : empty
| **in** query_identifier
| **in** node_set_identifier

conceptual_part : empty
| **with** components

components : component
| components boolean_operator component

component : field_name
| field_name relational_operator value

environment_part : empty
| **environment** connections

connections : connection
connections connection

connection : link_name target subset_part
| domain_predicate

subset_part : empty
| **subset** components

target : node_name
| { node_list }
| partition_part

node : class_name
| class_name:instance_name

node_list : node_name
| node_list , node_name